

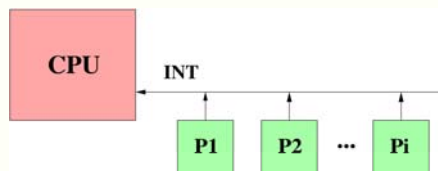
E/S por interrupciones

Operación con múltiples periféricos


Problemática

- Conexión
- Identificación del solicitante
- Localización de la rutina de servicio correspondiente
- Prioridades en caso de peticiones simultáneas
- Anidamiento de rutinas de servicio

Conexión de varios periféricos



- La línea forma un OR cableado:
 - Se activa en el momento que alguno la activa y permanece activa hasta que todos la desactiven



Identificación mediante muestreo

```


DRTI: PUSH    .R1
IN           .R1, /Dir_C/E_P1
CMP         .R1, #INT_Pendiente_P1
BZ          $Rut_Espec_P1
IN          .R1, /Dir_C/E_P2
CMP         .R1, #INT_Pendiente_P2
BZ          $Rut_Espec_P2
...

```

Rut_Espec_P1 Salvar estado
Código
específico
de P1
Restaurar estado
RETI

Rut_Espec_P2 Salvar estado
Código
específico
de P2
Restaurar estado
RETI
...


AC: Sistema de E/S 42



Análisis

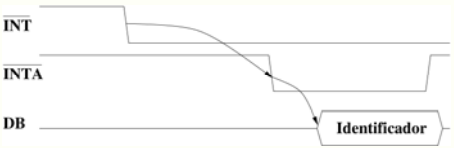
- En caso de peticiones simultáneas solo se ejecuta la más prioritaria que es la que se consulta primero
- No es posible el anidamiento entre las distintas rutinas de servicio porque la interrupción ha de ser previamente reconocida para averiguar su prioridad
- Se utilizan muchas instrucciones en la identificación de la interrupción cuando existen muchos periféricos

AC: Sistema de E/S 43

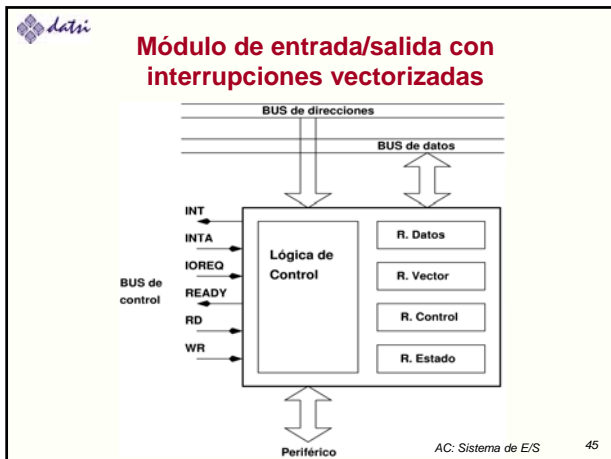


Vectorización

- Mediante el **ciclo de bus de reconocimiento de interrupciones** la CPU pide que el peticionario de la interrupción se identifique: **vector de interrupción**.
- El identificador se carga al iniciar la operación en el registro del vector de interrupción del módulo de E/S.



AC: Sistema de E/S 44



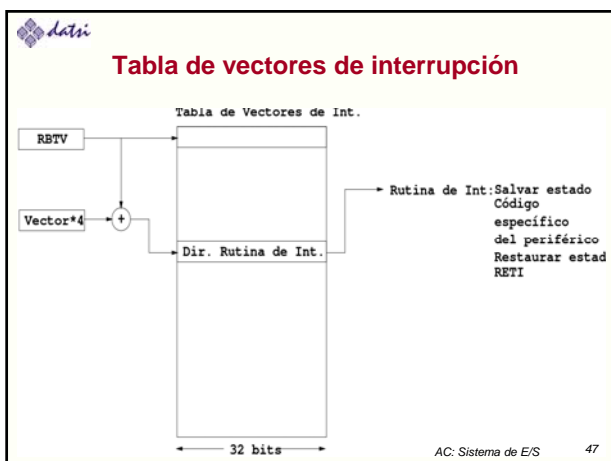
Secuencia de reconocimiento con vectorización

FETCH: Si $INT \wedge RE.\overline{BMI}$ entonces:

Salvar PC
 Salvar RE
 $RE.BMI \leftarrow 1$ (Inhibir interrupciones)
 $RE.S \leftarrow 1$ (Cambiar a modo privilegiado)
 Ciclo de bus de reconocimiento de interrupciones
 (activación \overline{INTA}) $DR \leftarrow \text{Vector}$
 $PC \leftarrow f(\text{Vector})$
 ir a *fetch*
 Si no
 ir a *fetch*

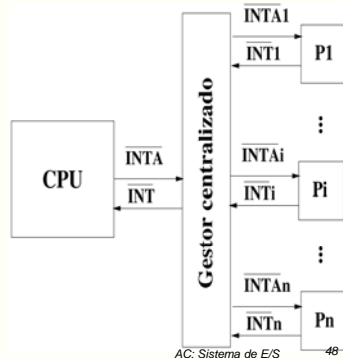
MUY MUY IMPORTANTE!!!

AC: Sistema de E/S 46



Prioridades

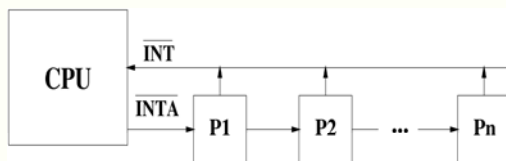
- La señal de reconocimiento le ha de llegar al más prioritario de los módulos solicitantes.
- Se necesita un esquema de prioridades hardware



AC: Sistema de E/S

48

Gestor encadenado (daisy chain)

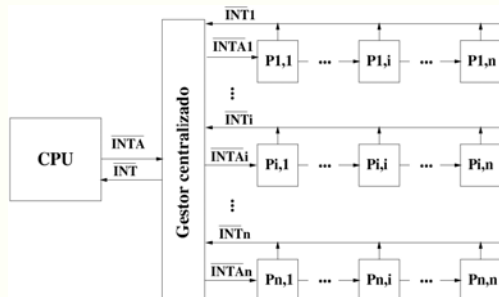


- El tiempo de ciclo de bus ha de ser suficiente
- Se pueden conectar tantos módulos como sea necesario

AC: Sistema de E/S

49

Híbrido



AC: Sistema de E/S

50

Análisis

- Rápida identificación del solicitante
- Conflictos de prioridades resueltos por hardware
- No es posible el anidamiento de rutinas de servicio:
 - Existe un solo biestable de máscara con lo que o están todas permitidas o inhibidas
 - Con varios biestables de máscaras de interrupción se puede implementar una **inhibición selectiva** por niveles:
 - nivel mínimo permitido
 - SRI: se prohíbe el nivel atendido y todos los inferiores

Secuencia de reconocimiento de interrupciones con inhibición selectiva

FETCH: Si $(i = \max(INT_j), \forall j) \wedge (i > RE.MI)$ entonces:

 Salvar PC

 Salvar RE

$RE.MI \leftarrow i$ (Establecer la nueva máscara de int.)

$RE.S \leftarrow 1$ (Cambiar a modo privilegiado)

 Ciclo de bus de reconocimiento de interrupciones (activación \overline{INTA}_i) $DR \leftarrow \text{Vector}$

$PC \leftarrow f(\text{Vector})$

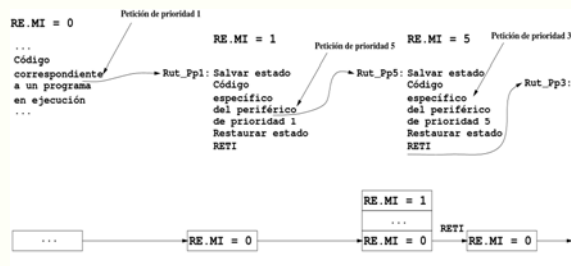
 ir a *fetch*

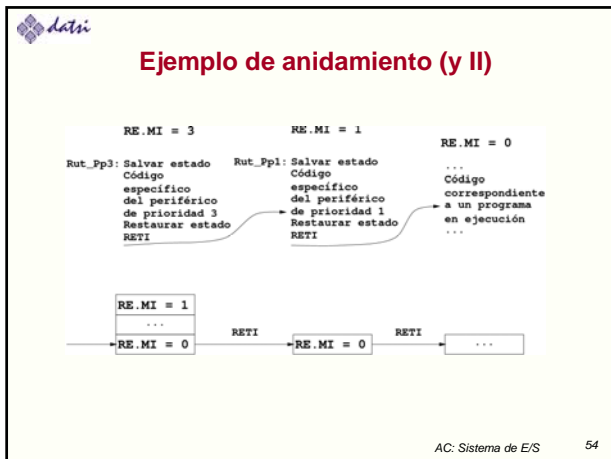
Si no

 ir a *fetch*

LIMUX: MUY IMPORTANTE!!!

Ejemplo de anidamiento

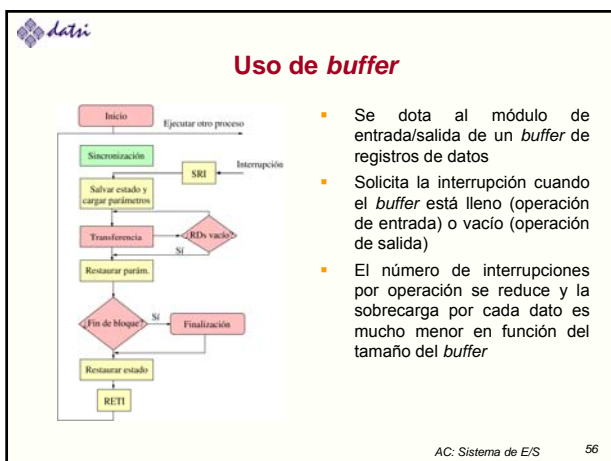




Análisis cuantitativo

- Se evita la sincronización pero se realizan otras operaciones para llevar a cabo la transferencia.
- El total supone mucho menos tiempo de CPU que por programa pero aún existe una sobrecarga inevitable.
- Para minimizar el impacto de esta sobrecarga se puede aumentar el tamaño del registro de datos.

AC: Sistema de E/S 55





Asignación de prioridades

- El método óptimo consiste en asignar mayor prioridad al dispositivo que pide interrupciones con mayor frecuencia: mantra: *"más prioridad al que solicita interrupción con más frecuencia"*.
- La frecuencia depende de la V_{transf} y del tamaño de Reg. de Datos/buffer.
- Algunos dispositivos excepcionalmente no siguen esta regla:
 - Consola de operación
 - Temporizadores programables
 - DMA

¡¡¡MUY IMPORTANTE!!!

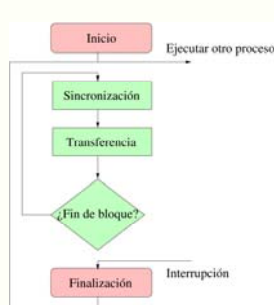


Interrupciones no "enmascarables"

- Existen sucesos que no admiten demora en su tratamiento.
- Por ejemplo un fallo de energía.
- Para ellos existe una línea especial de petición de interrupción que no puede ser enmascarada:
 - NMI en la familia x86
 - INT₇ en la familia M68000

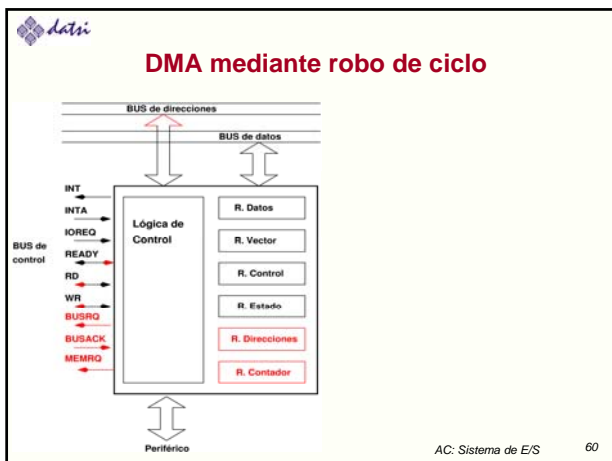


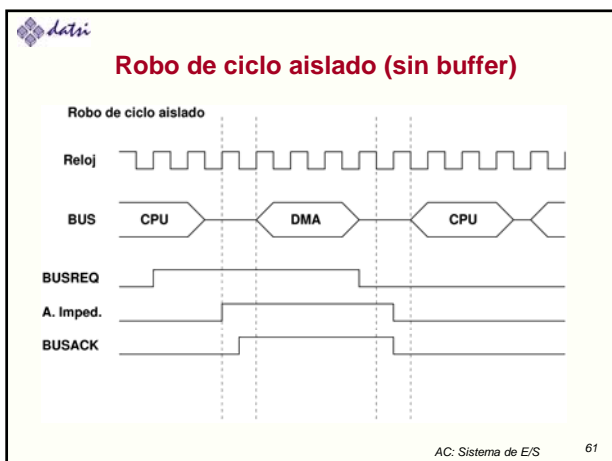
Entrada/salida mediante DMA

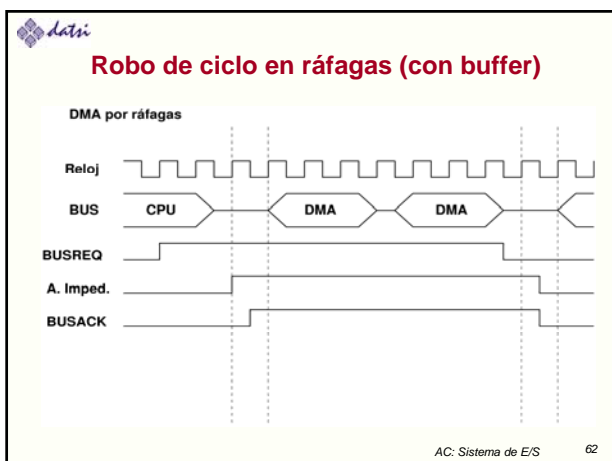



- La CPU se encarga de iniciar la operación.
- El módulo de E/S se encarga de la sincronización y transferencia y "avisa" cuando ha terminado.
- La CPU finaliza la operación.
- Hay una única interrupción por operación: se ahorra mucho tiempo de CPU con dispositivos de bloque.

¡¡¡MUY IMPORTANTE!!!










Operación por DMA

- Programar la operación de DMA en el periférico: instrucciones de salida (OUT)
 - Dir. Comienzo zona de Memoria: Dir. → `DIR`
 - Tamaño del bloque: Nº datos → `CONT`
 - E/S: E/S → `ES`
 - A partir de ese momento la CPU se dedicará a "sus labores"
- Cuando el **periférico** está listo para la transferencia de un dato, **solicita los buses**:
 - Activa **BUSREQ**
 - La CPU "cederá" los buses cuando acabe la "fase" actual de la instrucción que está ejecutando:
 - "Completar" la ejecución de la instrucción (no ejecuta la siguiente Fase):
 - Pone en "alta impedancia" sus conexiones a:
 - Bus de Datos (**DB**)
 - Bus de Direcciones (**AB**)
 - Señales de control de Memoria (**RD, WR, etc.**)
 - Activa **BUSACK**
 - El **periférico**:
 - Transfiere el dato al/desde M activando las ss. de control y dirección pertinentes y usando el Bus de Datos.
 - `CONT` ← `CONT` - 1
 - `DIR` ← `DIR` + 1
 - Si `CONT` = 0 ("falta más datos por transferir") entonces:
 - Si perif. listo para transferir siguiente dato entonces ("modo full duplex")
 - Mantiene activa **BUSREQ**
 - GO TO 3
 - Si no ("modo de datos atado")
 - Desactiva **BUSREQ**
 - la CPU
 - desactiva **BUSACK**
 - continúa ejecutando Fase siguiente
 - GO TO 2
 - Si no ("CONT" = 0: bloque de datos transferido")
 - Desactiva **BUSREQ**
 - "avisar" a la CPU de que ha terminado la operación de DMA: solicita interrupción.

Activa **IRQ**

AC: Sistema de E/S

63



E/S por DMA


Visión cuantitativa

$t_{op} \dots$
 $t_{CPU} \dots$

!!!MUY IMPORTANTE!!!

AC: Sistema de E/S

64



Diferencias y similitudes con las interrupciones

- Ambos son eventos asíncronos pero los robos de ciclo no alteran el estado de la CPU: **!!!NO SE EJECUTAN INSTRUCCIONES!!!**
 - Por lo tanto se puede conceder en "cualquier" momento.
- Un ciclo de bus no es reanudable y muy breve.
 - Por lo tanto no se permite anidamiento.
- Para ambos mecanismos existe una línea para petición y otra para concesión:
 - Para peticiones de bus simultáneas se utilizan los esquemas de prioridades hardware: centralizado, daisy chain o híbrido.

AC: Sistema de E/S

65
